Experimental results showed that this method is not sensitive to the value of k and doesn't need a specific k-nearest neighbor graph creating.

Bibliography

[Alpert, 1997] C. J. Alpert, J. H. Huang and A. B. Kahng, Multilevel circuit partitioning. In: Proc. of the 34th ACM/IEEE Design Automation Conference. 1997.

[Guha, 1999] S. Guha, R. Rastogi, K. Shim ROCK: Robust Clustering using linKs, (ICDE'99)

[Karypis, 1997] G. Karypis, R. Aggarwal, V. Kumar and Sh. Shekhar, Multilevel hypergraph partitioning: Application in VLSI domain. In: Proceedings of the Design and Automation Conference. 1997.

[Karypis, 1998] G. Karypis, and V. Kumar, hMETIS 1.5.3: A hypergraph partitioning package. Technical report. Department of Computer Science, University of Minnesota, 1998

[Karypis, 1999a] G. Karypis, E.-H. Han, and V. Kumar. CHAMELEON: A Hierarchical Clustering Algorithms Using Dynamic Modeling. IEEE Computer, 32(8):68–75, 1999.

[Karypis, 1999b] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In Proceedings of the Design and Automation Conference, 1999.

[Karypis, 2003] G. Karypis, CLUTO 2.1.1. A Clustering Toolkit. Technical report. Department of Computer Science, University of Minnesota, 2003

[Karypis lab.] http://www.cs.umn.edu/~karypis.

[Mitchell, 1997] T. M. Mitchell. Machine Learning. McGraw Hill, 1997

[Zhang, 1996] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH : an efficient data clustering method for very large databases. SIGMOD'96.

Authors' Information

Shatovska Tetyana – Kharkiv National University of Radio Electronics, Computer Science department, P.O.Box: Kharkiv-116, Lenin av. 14, Ukraine; e-mail: tanita_uk@mail.ru

Safonova Tetiana - Kharkiv National University of Radio Electronics, Computer Science department, P.O.Box: Kharkiv-116, Lenin av. 14, Ukraine; e-mail: safonovatm@mail.ru

Tarasov Iurii - Kharkiv National University of Radio Electronics, Computer Science department, P.O.Box: Kharkiv-116, Lenin av. 14, Ukraine; e-mail: ytarasovmail@rambler.ru

TECHNOLOGY OF STORAGE AND PROCESSING OF ELECTRONIC DOCUMENTS WITH INTELLECTUAL SEARCH PROPERTIES

Yuri Kalafati, Konstantin Moiseyev, Sergey Starkov, Svetlana Shushkova

Abstract: The technology of record, storage and processing of the texts, based on creation of integer index cycles is discussed. Algorithms of exact-match search and search similar on the basis of inquiry in a natural language are considered. The software realizing offered approaches is described, and examples of the electronic archives possessing properties of intellectual search are resulted.

Keywords: dynamic systems, associative search, integer index cycles, text indexation, archiving and retrieval.

Introduction

The use of dynamic systems for record and data processing was first offered in [Dmitriev etc., 1991]. The basic idea consists in the fact that a correspondence is put between a given set of information blocks and a set of limiting cycles of a discrete nonlinear dynamic system: one-dimensional [Dmitriev, 1991], [Andreyev etc., 1992], [Andreyev etc., 1997] or multidimensional [Andreyev etc., 1994] maps. The final sequence of symbols of a certain alphabet is meant by the information block and each symbol is put in conformity with the dynamic system

variable. This approach has been realized to record the text and graphic information by means of piecewise-linear maps and the scope for associative searching of information by its fragments is shown. The research has been developed further [Andreyev etc., 1999] and to optimize the search an integer index was suggested to be used instead of dynamic variables. In spite of the fact that the principles of information representation with the integer index cycles are well-known and successfully applied for data storage in various DB, the creation of electronic archives possessing the opportunities of intellectual search, including associative search, exact-match search, search similar to inquiries in the natural language is of special interest and is to be developed. Consider some key moments of a technology being developed.

Indexation algorithm

Assume, the first page presented by the next text line is transmitted for indexation:

a b c a c a d a b a c a b c (1)

As a first step, the symbol – an attribute of the beginning of a page is added to the initial alphabet. This symbol will have the number 256. Transform the initial text to a cyclic sequence of symbols and add a symbol 256 to the end of a sequence of symbols. To avoid excessive information, the concept of an expanded alphabet is entered in the initial sequence. Each element of an expanded alphabet is based on two already existing symbols. For example, if it is necessary to escape the repeated sequence of symbols a b c, a new symbol 257 = ‘ a ’ + ‘ b ’ and 258 = 257 + ‘ c ’ is entered and so the sequence a b c in the initial text is replaced with a symbol 258. The text (1) is indexed using the above concept. Let us search the repeated pairs of symbols in the initial text. Take the pair of symbols from a given example– the pair a b. As a given pair of symbols is found more than once in the text, we add this pair to the expanded alphabet:

Symbol 257 = ‘ a ’ + ‘ b ’

Transform the initial sequence of symbols, having replaced the pair symbols a b by a new symbol with the index 257. We have:

257 c a c a d 257 a c 257 c 256 (2)

Then find the recurrence of other pairs and create new symbols. Based on this sequence, create a page description array, each element of which consists of three symbols: a_n, a_{n+1}, a_{n+2} , and order the array obtained in a pair (a_n, a_{n+1}). A necessary condition for system operation is the uniqueness of a pair (a_n, a_{n+1}) in the array formed. The following actions are to be done in adding the next page:

The text transmitted for indexation is processed using the expanded alphabet formed earlier. First there is a search in the added text of a pair of symbols corresponding to the element 257 in the expanded alphabet. If such pairs are found, they are replaced by the symbol 257. Then there is a search of a pair of symbols corresponding to the element 258 and so on.

The repeated pairs of symbols are searched in the sequence of symbols obtained. If these are found, a new element of the expanded alphabet is created and the corresponding pairs are replaced with a new element.

Then each pair of symbols from the sequence created is being searched in the page description array available. If such a pair is found, a new symbol of the expanded alphabet is created and the corresponding pair of symbols in the added text is replaced by a new symbol. In the program realization available there also proceed the change in already existing array and the replacement of a pair of symbols with the newly created symbol. The description of this algorithm updating is beyond the scope of this paper.

Transform the obtained cyclic sequence to a set of elements for a page description array. Add these elements to an array and reorder it.

The expanded alphabet is presented as a two-dimensional integer array. An array index is the number of a symbol in the expanded alphabet. The first element of an array has the index 257. Each couple of integers stored in the array element are components of the corresponding alphabet symbol. The ordered page description array is the array consisting of elements (a_n, a_{n+1}, a_{n+2}). This array is ordered in a pair (a_n, a_{n+1}) and this pair is unique within the whole indexed text *sets.* Input points *on indexed pages.* Any pair of symbols of each page representation is stored as a two-dimensional array of integers. Array index is the number of the page indexed, and the array element content is the information necessary and sufficient to begin the procedure of extracting corresponding text page.

Text extraction from the page

To extract the text page from the constructed index it is sufficient to have information about any pair of symbols in the expanded alphabet (a_{n0}, a_{n0+1}) contained in the page. It should be remembered that the page description array is ordered in the first pair (a_n, a_{n+1}) and the condition required. i.e. the uniqueness of this pair in the whole file index, is fulfilled. Using the procedure of binary search we are searching the pair (a_{n0}, a_{n0+1}) in a page description array. From the element found in a file we take an element a_{n0+2}. Then in the page description array we search the pair (a_{n0+1}, a_{n0+2}). From the element found in a file we take the element a_{n0+3} and so on. We repeat this operation until after search of the pair (a_i, a_{i+1}) and extraction of the element a_{i+2} the equality: $a_{i+1} = a_{n0}$ and $a_{i+2} = a_{n0+1}$ is true.

Thus, we have obtained the cyclic sequence describing the page text in symbols of the expanded alphabet. Then based on the procedure of decoding the symbol of then expanded alphabet (see below) we transform the cyclic sequence presented in symbols of the expanded alphabet in a sequence of ASCII symbols.

Description of base algorithm search

To demonstrate the principal opportunity of information search with a given way of information indexation the following mechanism is suggested. A sentence or even a text paragraph is taken as an input for searching. The search inquiry is coded by means of the expanded alphabet available. Assume that the search inquiry was transformed to the sequence $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8$ of elements of the expanded alphabet. Take the first pair of the search inquiry (a_1, a_2) and search for it in the ordered page description array. If such a pair of symbols is not found, pass to the pair of symbols (a_2, a_3) and so on. In case of a successful search, for example, of the pair (a_2, a_3), we have (a_2, a_3, a_x) an element of the page description array. Compare the element a_4 and the element a_x. If these elements coincide, find (a_3, a_4, a_y) the element of a page description array. If elements a_5 and a_y coincide, the search is considered to be successful.

Two versions of search are realized based on the suggested indexation algorithm of text information:

Exact-match search

A given version of search is intended for solving a problem of search by a word or by a short search inquiry. It allows to find all inputs of search inquiry in the indexed text array. According to the offered algorithm of indexation there are some possible variants for line representation corresponding to the search inquiry inside the indexed file. First, the whole line of a search inquiry can be entered inside a symbol of the expanded alphabet. All such symbols as well as symbols created on their basis are to be searched in a page description array. Each found element of a page description array which is the input point to one of the pages of the file indexed is considered to be the result of search.

Second, the line, corresponding to the search inquiry in the indexed page, can be divided into two parts, one of which is the end and the second is the beginning of one of symbols of the expanded alphabet. To find all such results of search we should construct two files of symbols. The first one is an array of all symbols of the expanded alphabet terminating with a given line of ASCII symbols. The second one is an array of all symbols of the expanded alphabet beginning with a given line of ASCII symbols. The corresponding pair of symbols should be searched for each element of the first array and each element of the second array in the page description array. In successful search the found element of a page description array is a result of search. Then the required line can consist of three parts. The first part is the end and the third one is the beginning of one of the symbols of the expanded alphabet. The second part is one of the elements of the expanded alphabet. As in the previous case, compile two files of symbols for the first and third parts of the search inquiry. For each symbol of the first part we create a pair of symbols formed by an element of this array and a symbol of the expanded alphabet corresponding to the second part of the search inquiry and search this pair in a page description array. If search is successful, search the third symbol of a file element in the file of symbols constructed for the third part of search inquiry. In case of success the found element of a page description array is considered to be the result.

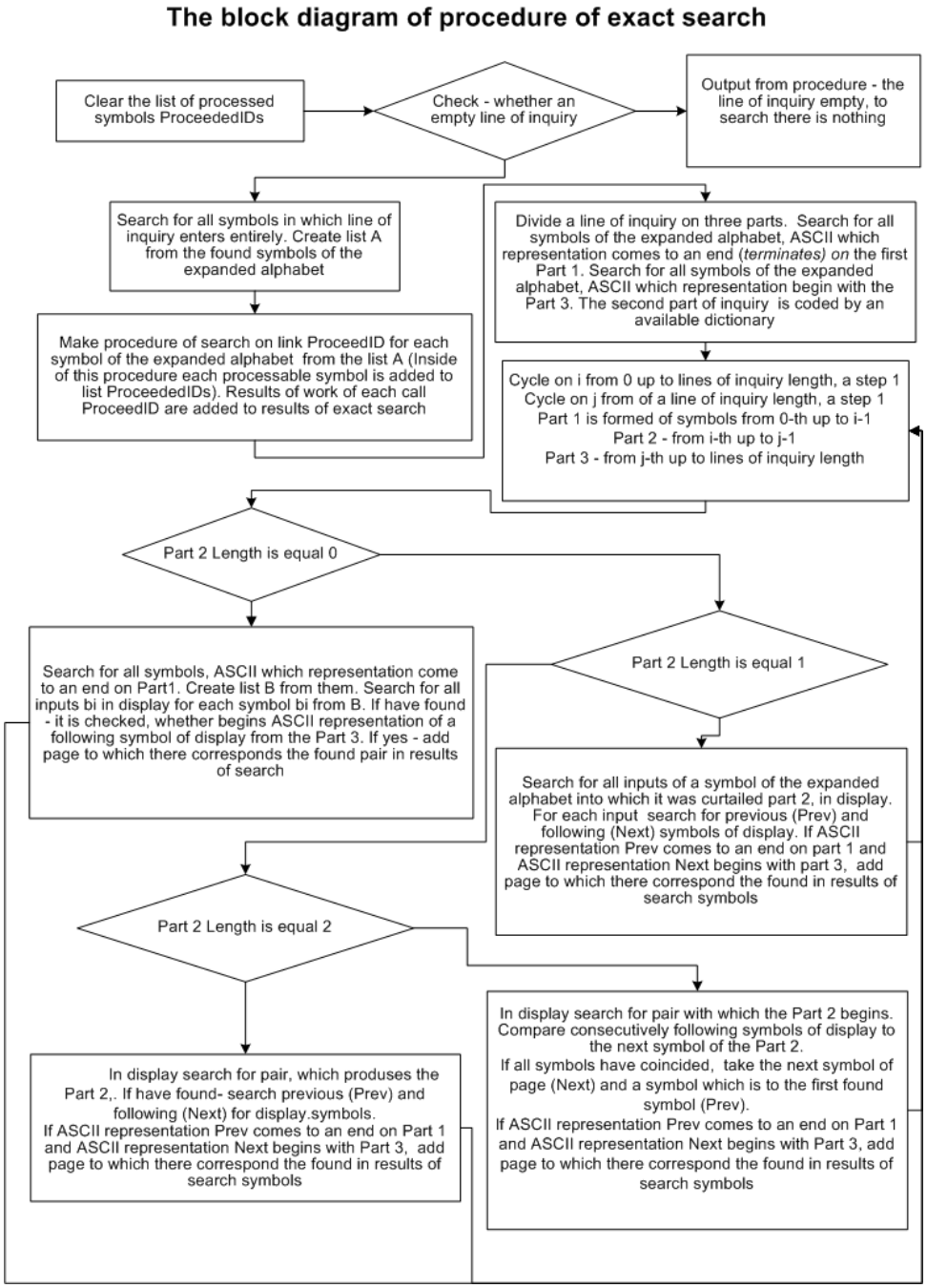

Fig. 1. Procedure of exact-match search.

Finally, if the number of parts by which a desired line in the indexed page is presented exceeds three, we can search the symbols displaced in the centre of the line. Assume that the line is presented by five parts. The first and fifth parts are, accordingly, the end and the beginning of symbols of the expanded alphabet. Each of the remaining parts is presented by one of the symbols of the expanded alphabet. First, we search for the pair of the symbols formed by the second and third parts. If this part is found in a page description array, the third symbol of this array element can be compared with the fourth part. If they coincide, a pair of symbols formed by the third and the forth parts is being searched in a page description array. Take the third element from the found element of a file and we search it in the fifth part, i.e. in a file of symbols beginning from with the end of search inquiry. If search is successful, it is necessary to compare only a part of the found page which is before the second part of a line with the first part. In case of success we have the result of search. Fig.1 presents a block-diagram of the exact search procedure.

Search of similarity

This variant of search is meant for finding information closest to the search inquiry. Offer in the natural language, a paragraph or even the whole page of the text can be transmitted as the search inquiry. The search inquiry transferred to the input of search of similar is coded by means of the expanded alphabet available.

On the basis of a list of symbols for each indexed page the following sum is calculated:

$$P_i = \sum_{k=1}^{N} (length_k)^{\alpha} * (count_k)^{\beta} ,$$

where $length_k$ is the length in ASCII symbols of the k-th element in a list of symbols, $count_k$ is the quantity of the k-th element in a list in page i, and are the external parameters. Then, the obtained Pi values are ordered and pages with the highest values are given to the user as results of search.

Conclusion

Now a described algorithm of text processing and algorithms of text-through search are realized and used in CCT Archive and CCT Publisher Companies Controlling Chaos Technologies software products. Software products are intended for the creation of electronic archives of not structured documents with an opportunity of text-through information search, and for creation and preparation for CD and DVD electronic books, encyclopedias, archives of magazines. Examples of successful application of software products are the electronic archives of magazines « Chemistry and the Life », "Quantum", "Knowledge-force".

Fig. 2. gives results of search system operation with electronic archive of magazine " Quantum " as an example. At the upper left is inquiry in the natural language on which the search was carried out, below is the ranged list of the documents found. To the left is the document page with the allocated inputs.

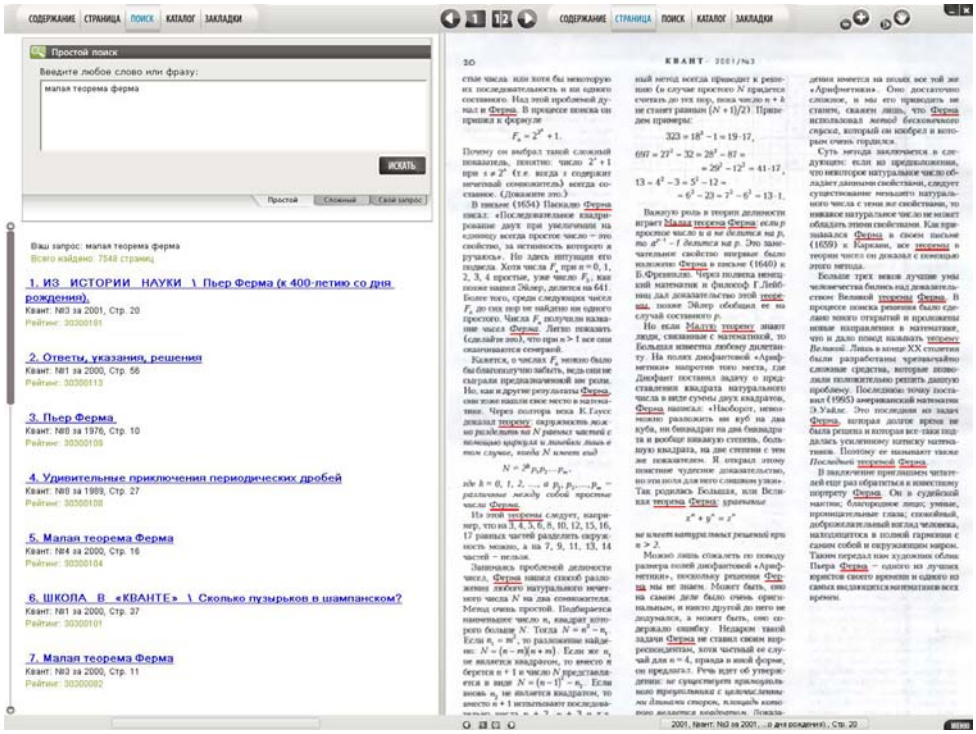

Fig. 2. Search system operation with electronic archive as an example

Below are the basic time characteristics managed to be reached with the present program realization of the algorithms described. All values are obtained using an ordinary personal computer, by the text size we mean the number of ASCII symbols in a text but not the size of files containing this text.

The maximal size of the indexed text is about 100 Mb

Text indexation rate is about 1 Mb per min (the average indexation rate 100 Mb)

Time of index opening is not more than 1 min.

Search time is about 1 sec.

It should be noted that the technology being developed is not language dependent and can be adjusted to any language systems. Development of ideas put in searching the similar allows one to solve such problems as search of plagiarism, *rubrication* and *text clusterization*, Internet content filtration and anti-spam system creation.

Bibliography

[Dmitriev etc., 1991] Dmitriev A.S., Panas A.I., Starkov S.O. Storing and recognition information based on stable cycles of 1-D map // Phys. Lett. A.–1991–. V.155, –№8–pp.494-499.

[Dmitriev, 1991] Dmitriev A.S. Record and recognition of information in one-dimensional dynamic systems // РЭ. –1991–V.36– №1–p. 101-108.

[Andreyev etc., 1992] Andreyev Y.V., Dmitriev A.S., Chua L.O., Wu C.W. Associative and random access memory using one-dimensional maps // IJBC–1992–V.3–№3–pp.483-504.

[Andreyev etc., 1997] Andreyev Yu. V., Dmitriev A.S., and Starkov S.O. Information Processing in 1-D Systems with Chaos, *IEEE Transactions on Circuits and Systems*, 1997, vol. 44, No. 1, pp. 21-28.

[Andreyev etc., 1994] Andreyev J.V., Belskij J.L., Dmitriev A.S. Record and restoration of the information with the application of steady cycles of two-dimentional multivariate displays // РЭ–1994–V.39–№1–p.114-123.

[Andreyev etc., 1999] Andreyev Yu. V., Dmitriev A.S., and Ovsyannikov A.V. Information searching system "Forget-Me-Not" based on complex dynamics of nonlinear systems, *Proc. 7th Int. Workshop NDES-99*, 1999. Ronne, Denmark. pp.273-276.

Authors' Information

Kalafati Yuri Dmitrievich – Senior Scientist, PhD, Institute or Radio Engineering & Electronics, RAS, Moscow; e-mail: kalafati@controlchaostech.com

Moiseyev Konstantin Vladimirovich – Director, Controlling Chaos Technologies, Moscow; e-mail: moiseyev@controlchaostech.com

Starkov Sergey Olegovich – Professor, Obninsk State Technical University of Nuclear Power Engineering, Kaluga region, Obninsk; email: Starkov@iate.obninsk.ru

Shushkova Svetlana Alexandrovna – Undergraduate, Obninsk State Technical University of Nuclear Power Engineering, Kaluga region, Obninsk; e-mail: shushkovasvetlana@ramler.ru

COMMON SCIENTIFIC LEXICON FOR AUTOMATIC DISCOURSE ANALYSIS OF SCIENTIFIC AND TECHNICAL TEXTS *

Elena Bolshakova

Abstract: The paper reports on preliminary results of an ongoing research aiming at development of an automatic procedure for recognition of discourse-compositional structure of scientific and technical texts, which is required in many NLP applications. The procedure exploits as discourse markers various domain-independent words and expressions that are specific for scientific and technical texts and organize scientific discourse. The paper discusses features of scientific discourse and common scientific lexicon comprising such words and expressions. Methodological issues of development of a computer dictionary for common scientific lexicon are concerned; basic principles of its organization are described as well. Main steps of the discourse-analyzing procedure based on the dictionary and surface syntactical analysis are pointed out.

* The work is supported by the grant № 06-01-00571 of Russian Fond of Fundamental Researches (RFFI).